REMARKS

The above Amendments and these Remarks are in reply to the Office Action mailed March 5, 2009.

Currently, claims 1-3, 5-15, 17-24, 26-30, 32-35, 37-42 and 44-49, 51-54 are pending. Applicants have amended claims 2-3, 5, 13-15, 17, 22-24, 26, 32, 35, 37, 39, 41-42, 47, 49, and 52. Claims 55-59 have been added. No new matter has been added as a result of these claim amendments. Applicants respectfully request reconsideration of claims 1-3, 5-15, 17-24, 26-30, 32-35, 37-42 and 44-49, 51-54 and consideration of new claims 55-59.

Objection to Specification

The disclosure is objected to because of the following informalities: term "non-synthetic" recited in claims 2, 5, 14, 17, 23, 26, 34, 37, 41, 47, 48 and 52 have not been defined in the specification.

Applicants have amended claim 2 as follows:

> The process according to claim 1, wherein:
> said step of determining whether to modify said method includes determining whether said method is a [[non-]]synthetic method; and
> said step of modifying includes modifying said method if said method is not a [[non-]]synthetic method and said method calls another method.

Note that the determining step now recites determining whether the method is a synthetic method. The instant specification in paragraph 37 defines a synthetic method as, "a method that does not appear in the source code." Therefore, the term "synthetic method" is clearly defined in the instant Specification. Moreover, one of ordinary skill in the art would have no difficulty understanding the scope of the claimed invention. Note that the term "synthetic method", as defined in the Specification in paragraph 37, in not limited to the JAVA programming language. Rather, JAVA is one example of a programming language that uses synthetic methods.

The modifying step now recites, "modifying said method if said method is not a synthetic method and said method calls another method." The meaning of the phrase, "if said method is not a synthetic method" is clearly understood to refer to the case in which the determination step

determines that the method is not synthetic. One of ordinary skill in the art would have no difficulty understanding the scope of the claimed invention.

Applicants have made amendments similar to those in claim 2 in other claims. Applicants respectfully request removal of the objection.

Claim Objections

Claim 32 is objected because the numbering of claims is not in accordance with 37 C.F.R. §1.126 which requires the original numbering of the claims to be preserved throughout the prosecution.

Claim 32 has been amended to change its dependency from claim 31 to claim 32. Therefore, the objection is believed to have been overcome.

Rejection of Claims 2, 3, 5, 14, 15, 17, 23, 24, 26, 34, 37, 41, 42, 47, 48 and 52 Under 35 U.S.C. §112

Claims 2, 3, 5, 14, 15, 17, 23, 24, 26, 34, 37, 41, 42, 47, 48 and 52 have been rejected under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Applicants have amended claim 2 as follows:

> The process according to claim 1, wherein:
> said step of determining whether to modify said method includes determining whether said method is a [[non-]]synthetic method; and
> said step of modifying includes modifying said method if said method is not a [[non-]]synthetic method and said method calls another method.

Note that the determining step now recites determining whether the method is a synthetic method. The instant specification in paragraph 37 defines a synthetic method as, "a method that does not appear in the source code." Therefore, the term "synthetic method" is clearly defined in the instant Specification. Moreover, one of ordinary skill in the art would have no difficulty understanding the scope of the claimed invention. Note that the term "synthetic method", as defined

in the Specification in paragraph 37, in not limited to the JAVA programming language. Rather, JAVA is one example of a programming language that uses synthetic methods.

The modifying step now recites, "modifying said method if said method is not a synthetic method and said method calls another method." The meaning of the phrase, "if said method is not a synthetic method" is clearly understood to refer to the case in which the determination step determines that the method is not synthetic. One of ordinary skill in the art would have no difficulty understanding the scope of the claimed invention.

Claim 3 has been amended to recite:

> The process according to claim 1, wherein:
>
> said step of determining whether to modify said method includes determining ~~whether said method has~~ an access level said method ~~of public or package~~; and
>
> said step of modifying includes modifying said method if said method has [[said]] an access level that satisfies a criterion ~~of public or package~~ and said method calls another method.

Because claim 3 no longer recites limitations regarding an access level being "public or package," the rejection to claim 3 has been overcome. Applicants respectfully assert that there is clear support in the instant specification for the claim limitations of, "determining an access level said method." Moreover, there is clear support for the claim limitations, "modifying includes modifying said method if said method has said access level that satisfies a criterion." Applicants respectfully assert that broad claim language should not be confused with language that fails to define the scope of the claims.

Applicants have amended claim 4 to recite, "determining whether said method is a synthetic method and has an access level of public or package in the JAVA programming language." Therefore, the rejection to claim 4 is believed to have been overcome.

Other claims containing the language such as, "method and has an access level of public or package" have been amended along the lines of either claim 3 or claim 4. Therefore, the rejection of remaining claims with such limitations is believed to have been overcome.

It is respectfully requested that the rejection of claims 2, 3, 5, 14, 15, 17, 23, 24, 26, 34, 37, 41, 42, 47, 48 and 52 under 35 U.S.C. §112, second paragraph, be withdrawn.

Rejection of Claims 1-3, 5-8, 10, 12-15, 17, 18, 20, 39-42, 44 and 47-54 Under 35 U.S.C. §103(a)

Claims 1-3, 5-8, 10, 12-15, 17, 18, 20, 39-42, 44 and 47-54 have been rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,351,843 to Berkley (hereinafter "Berkley") in view of U.S. Patent No. 6,738,965 to Webster (hereinafter "Webster") in further view of "Call Graph Construction in Object-Oriented Languages to Grove et al. (hereinafter "Grove").

Currently amended claim 13 recites:

> A process for monitoring, comprising:
>
> determining which methods of a set of methods satisfy criteria for likely being at the top of a call graph without using information from a call graph; and
>
> using a first tracing mechanism for said methods that satisfy the criteria without using said first tracing mechanism for methods that do not satisfy the criteria.

Support for the amendment to claim 13 may be found in the instant specification at least in paragraphs 32-35. As described in the instant Specification in paragraph 34, determining which methods are at the top of the call graph can, in some situations, be very difficult. In paragraph 35, an example set of criteria is provided for determining whether the method is likely to be at the top of a call graph. Note that the technique does not require the use of information from a call graph.

Applicants respectfully assert that Berkley in view of Webster in further view of Grove fails to disclose, "determining which methods of a set of methods satisfy criteria for likely being at the top of a call graph without using information from a call graph," as claimed. Further, Berkley in view of Webster in further view of Grove fails to disclose, "using a first tracing mechanism for said methods that satisfy the criteria without using said first tracing mechanism for methods that do not satisfy the criteria," as claimed.

Berkley is concerned with inserting functions into an existing application executable of an object-oriented computer system without recompiling the executable. Berkley modifies runtime configuration settings to add a setting that specifies the function for at least one class of the

application executable. Berkley runs the application executable using the modified configuration settings, and if it is determined that a function is active for a class then a re-direction stub is created dynamically to implement the function for the class (col. 2, lines 18-29).

Berkley discusses that in an object oriented system it can be difficult to determine the flow of the program. In this discussion, Berkeley refers to a call graph and the complexities that arise when different methods call each other. Specifically, Berkley states that if an error occurred in a "current" method, then the execution of all "previous" methods is important to know the state that an object was in when the current method was called (col. 1, lines 43-62).

What the foregoing clearly indicates is that Berkley is simply not interested in, "determining which methods of a set of methods satisfy criteria for likely being at the top of a call graph without using information from a call graph," as claimed. Nor would one of ordinary skill in the art have any reason to modify Berkley to determine which methods are at likely to be at the top of the call graph (and modify only those methods). To do so would not provide Berkley with the information such as, "the state on object was in when the current method was called."

Therefore, even if a reference such as Grove discloses constructing a call graph, one would not be motivated to use information from the call graph to modify Berkley to arrive at the limitations of claim 13. Moreover, claim 13 recites that information from a call graph is not used is determining which methods satisfy criteria for likely being at the top of a call graph without using information from a call graph. Therefore, Grove's teachings are not pertinent. Note that claim 13 recites an invention that overcomes problems associated with determining which methods are at the top of the call graph.

The Office Action appears to reason that Webster discloses that a user can specify which methods to trace according to some purpose or interest of the user. However, Webster does not teach tracing a method that is determined to likely be at the top of a call graph. Moreover, it certainly would not be obvious to go from the alleged teaching in Webster that one can trace a method interest to tracing a method that is determined to likely be at the top of a call graph. There simply is no guidance at all in Webster, or other references, for determining that a method that is likely at the top of the call graph should be traced.

For all of the foregoing reasons, claim 13 is allowable.

Claim 1 recites:

> A process for monitoring, comprising:
> accessing a method;
> **determining whether to modify said method, said step of determining whether to modify said method includes determining whether said method calls another method**; and
> modifying said method for a particular purpose if said method calls another method.

Applicants respectfully maintain that the prior art fails to disclose modifying a method for a particular purpose if the method calls another method. Neither Berkley nor Webster nor Grove, alone or in combination, disclose "said step of determining whether to modify said method includes determining whether said method calls another method," as claimed.

Berkley fails to disclose "said step of determining whether to modify said method includes determining whether said method calls another method," as claimed. The Office Action appears to concede that Berkley fails to disclose determining whether a method calls another method. However, the Office Action also asserts the Berkley discloses a "call stack" which **can be used** to determine a method that calls other methods (page 4, line 12-15). However, the mere fact that a call stack could be used to determine which methods call other methods in no way teaches **determining which methods to modify** based on which methods call another method. Berkley discloses a technique for inserting a function into an existing application executable (col. 2, lines 18-19), which the Office Action appears to interpret as modifying a method. However, Berkley determines where to insert the function by checking configuration settings (col. 2, lines 17 – 55) not by "determining whether said method calls another method." Therefore, Berkley does not teach these claim limitations.

The Office Action appears to allege that modifying Webster overcomes this deficiency in Berkley. First Applicants assert that Webster fails to disclose "said step of determining whether to modify said method includes determining whether said method calls another method," as claimed. Webster discloses that a user can specify "methods of interest" to trace. Webster discloses that the user can specify those methods in a file (col. 5, lines 52-57). Webster discloses that the methods can

be specified either by class or method name (col. 6, lines 5-7). However, merely stating that a method that a user desires to trace can be specified by method name does not equate to disclosing determining to modify a method based on whether the method calls another method. In other words, Webster does not disclose anything about determining which methods to modify much less modifying a method based on whether the method calls another method. Thus, even Gove discloses constructing a call graph, there is no guidance in the prior art for using the information from the call graph to determine which methods to modify.

The Office Action appears to reason that Webster discloses that a user can specify which methods to trace according to some purpose or interest of the user. Further, the Office Action appears to assert that it is well known in the computer arts that one such purpose or interest of the user is methods calling other methods (page 4, lines 6-9 and page 9, lines 2-4). Applicants assert that it is not sufficient to allege that it is well known in the art that a method may call another method and therefore it **is of interest to modify the calling method**. That is, just because someone of ordinary skill in the art is aware that one method calls another method does not mean that it is well known in the art that it would be of interest to modify the calling method. In other words, there is a large gap between knowing that a method calls another method and deciding to modify the calling method because it calls another method. Applicants respectfully assert that it is improper to attempt to fill in this gap by taking "Official Notice." Applicants respectfully assert that **the Office Action presents no documentary evidence** to support that it is well known in the computer arts to determine whether to modify a method based on whether or not a method calls another method.

Thus, the Applicants respectfully assert that the Office Action is improperly relying upon Official Notice. As stated in MPEP 2144.03 "[o]fficial notice unsupported by documentary evidence should only be taken by the examiner where the facts asserted to be well-known, or to be common knowledge in the art are capable of **instant and unquestionable demonstration** as being well-known." As noted by the court in *In re Ahlert*, 424 F.2d 1088, 1091, 165 USPQ 418, 420 (CCPA 1970), the notice of facts beyond the record which may be taken by the examiner must be "capable of such instant and unquestionable demonstration as to defy dispute" (citing *In re Knapp Monarch Co.*, 296 F.2d 230, 132 USPQ 6 (CCPA 1961)). Applicants respectfully assert that it is not instantly and unquestionably known to modify a method based on whether or not the method calls

another method. Furthermore, as noted by the court in *Ahlert*, any facts so noticed should be of notorious character and serve only to "fill in the gaps" in an insubstantial manner which might exist in the evidentiary showing made by the examiner to support a particular ground for rejection. It is never appropriate to rely solely on common knowledge in the art without evidentiary support in the record as the principal evidence upon which a rejection was based. See *Zurko*, 258 F.3d at 1386, 59 USPQ2d at 1697; *Ahlert*, 424 F.2d at 1092, 165 USPQ 421. Applicants respectfully assert that the Office Action is not merely "filling in the gaps" in an **insubstantial** manner. To the contrary, the documentary evidence that is missing is **substantial**.

Furthermore, Berkley discloses monitoring classes in an object-oriented environment and **cannot be modified** to trace on a method by method basis. Berkley discloses monitoring classes in an object-oriented environment. A class may include one or more objects, and the class defines how an object is implemented. Objects are instances of classes (col. 4, lines 61-62). Each object may include one or more methods, as shown in Figure 1. Therefore, a class may contain one or more objects, which each may contain one or more methods. As shown in Figure 1, a method may or may not call another method. For example, in Object 1, which is part of a class, Method A calls Method B. However, Method D does not call any other method.

Berkley discloses monitoring methods by class as follows. If a user wishes to monitor a particular method within a program, the user can change configuration settings associated with the program to specify or activate the class of the method that the user wishes to monitor. "[C]onfiguration settings 300 received as part of a program's execution code are modified at runtime to add a setting to specify (as one example) tracing for a desired class of the executable 310, thereby producing new configuration settings 320" (Berkley Figure 5 and col. 6, lines 52-53). Berkley further explains that:

> the new configuration settings 330 are then employed with the existing application executable 340 to run the application executable 350. The object runtime will query the configuration settings and when a trace is active for a given class, will dynamically insert the trace class into the inheritance hierarchy for that class… Runtime will then see the trace class within the hierarchy and setup for a trace 360 (Berkley col. 6, line 64 – col. 7, line 4).

For example, for a class named 'class 1' that is specified in the configuration settings, the trace is set up through "redirection stubs [that] are created [to] trace entry and exit methods around each target method within class 1" (Berkley col. 7, lines 61-63). A trace is active for a given class when the user specifies the class in the configuration settings. The example for activating classes in column 7, lines 20-26 of Berkley shows:

> configuration settings used to set up a desired tracing environment. A configuration variable is initialized with the name(s) of the class(es) whose methods should be traced:
> [TraceOptions]
> ClassTrace = Class1, Class2, Class3, Class4

In the above example, Class1, Class2, Class3, and Class4 are set active by specifying these classes in the ClassTrace list. **All of the methods** in these classes will be traced. Using Figure 1 of Berkley as an example, if Object 1 was contained within Class 1, which is specified in the ClassTrace list above, all of the methods contained within Object 1 would be modified for tracing, even Method D which does not call any other method. Since all of the methods are traced, Berkley cannot be modified to arrive at the claimed invention.

As the foregoing clearly states, Berkley discloses that tracing performed by activating **classes**. Berkley does not disclose a technique for a user to specify a particular method to trace. Therefore, Berkley cannot be modified to trace individual methods.

For all of the foregoing reasons, claim 1 is patentable over the cited prior art.

Claim 39 recites:

> An apparatus capable of monitoring, comprising:
> means for determining whether a method calls another method;
> means for determining whether said method can be called by a sufficient scope of one or more other methods;
> means for determining whether said method is not a synthetic method; and
> means for tracing said method for a particular purpose only if said method calls another method, said method can be called by a sufficient scope of one or more other methods, and said method is not a synthetic method.

Claim 39 recites an embodiment in which tracing a method is only performed if three conditions are met: 1) if said method calls another method, 2) if said method can be called by a sufficient scope of one or more other methods, and 2) if said method is not a synthetic method. Applicants respectfully assert that **none** of these three conditions are taught or suggested in the prior art, much less the combination of the three. Applicants will discuss the three conditions separately for purposes of discussion, but not that all three conditions must be met for claim 39.

For at least the reasons discussed in the response to claim 1, the prior art fails to teach, "means for tracing said method for a particular purpose only if said method calls another method…"

For the following reasons, neither Berkley nor Webster nor Grove, alone or in combination, disclose "means for tracing said method for a particular purpose only if said method … is not a synthetic method." Even if, for the sake of argument, that it is true that one of ordinary skill in the art knows how to determine whether a method is not a synthetic method, this does not in any way mean that one of ordinary skill in the art would be motivated to trace only those methods that are not a synthetic method.

Applicants respectfully disagree with the allegation that appears at least on page 7 of the Office Action that, "[o]ne [of ordinary skill in the art] would have been motivated to do so [that is to determine whether a method is synthetic] to **allow a user to trace specified one or more methods** of which the function is to be implemented as suggested by Berkley." Applicants respectfully assert that the Office Action **may not look to the Applicants' invention for reasons (or motivations) to modify the prior art.** To do so clearly indicates that the Office Action is impermissibly using the Applicants' disclosure as a blueprint for modifying the prior art to attempt to arrive at the claimed invention. **It is the Applicants who have taught that one condition for modifying methods is that the method not be synthetic.** The Office Action **must not use the Applicants' own teaching against them.** Based on the foregoing, no reason (other than those taught by the Applicant) has been given for modifying those methods that are not synthetic.

For the following reasons, Neither Berkley nor Webster nor Grove, alone or in combination, disclose "means for tracing said method for a particular purpose only if said method … can be called by a sufficient scope of one or more other methods." Even if, for the sake of argument, that it is true

that one of ordinary skill in the art knows how to determine whether a method can be called by a sufficient scope of one or more other methods, this does not in any way mean that one of ordinary skill in the art would be motivated to trace only those methods that can be called by a sufficient scope of one or more other methods.

Applicants respectfully disagree with the allegation that appears at least on page 8 of the Office Action that, "[o]ne [of ordinary skill in the art] would have been motivated to do so [that is to determine whether a method has an access level of public or package] to **allow a user to trace specified one or more methods** of which the function is to be implemented as suggested by Berkley." Applicants note that whether a method has an access level of public or package is one way, but not the only way, to determine that a method can be called by a sufficient scope of one or more other methods. Applicants respectfully assert that the Office Action **may not look to the Applicants' invention for reasons (or motivations) to modify the prior art.** To do so, clearly indicates that the Office Action is impermissibly using the Applicants' disclosure as a blueprint for modifying the prior art to attempt to arrive at the claimed invention. **It is the Applicants who have taught that one condition for modifying methods is that the method can be called by a sufficient scope of one or more other methods.** The Office Action **must not use the Applicants' own teaching against them.** For the foregoing reasons, there in fact has been no reason given (other than those taught by the Applicant) for modifying those methods that can be called by a sufficient scope of one or more other methods.

For all of the foregoing reasons, claim 39 is allowable.

Claim 40 recites:

> An apparatus capable of monitoring, comprising:
> a storage device; and
> one or more processors in communication with said storage device, said one or more processors perform a process comprising:
> accessing a method,
> determining whether said method calls one or more different methods and can be called by a sufficient scope of one or more other methods, and
> tracing said method for a particular purpose if said method calls one or more different methods and can be called by a sufficient scope of one or more other methods.

Claim 40 is allowable for at least the reasons discussed in the response to claim 1. Claim 40 is also allowable for at least the reasons discussed in the response to claim 39 that pertain to the limitations, "tracing said method for a particular purpose if said method calls one or more different methods and can be called by a sufficient scope of one or more other methods."

Claim 47 recites:

> A process for monitoring, comprising:
> accessing a method;
> determining whether said method is complex, said step of determining includes determining that said method is complex if said method satisfies at least one of the following criteria:
> said method calls another method;
> said method has an access level that satisfies a criterion; or
> said method is not flagged by a compiler as being synthetic; and
> adding a tracer to said method only if said method is determined to be complex.

For reasons discussed in the response to claim 1, neither Berkley nor Webster nor Grove, alone or in combination, disclose "said step of determining includes determining that said method is complex if said method satisfies at least one of the following criteria … said method calls another method… adding a tracer to said method only if said method is determined to be complex," as claimed.

Neither Berkley nor Webster nor Grove, alone or in combination, disclose "said step of determining includes determining that said method is complex if said method satisfies at least one of the following criteria … said method has an access level that satisfies a criterion … adding a tracer to said method only if said method is determined to be complex," as claimed. The Office Action asserts that it is well known that the Java specification defines flags that indicate the access level of a method. Even if this is true, there is no teaching in the prior art **to determine that a method is complex if said method has an access level that satisfies a criterion and to add a tracer to said method only if said method is determined to be complex**. Thus, there is more missing from the teaching of the prior art than the Office Action concedes. Additionally, for reasons discussed in the response to claim 1, Berkley cannot be modified to add a tracer to said method only if said method is

determined to be complex for the case in which the criteria for a method being complex is if said method has an access level that satisfies a criterion.

Neither Berkley nor Webster nor Grove, alone or in combination, disclose "said step of determining includes determining that said method is complex if said method satisfies at least one of the following criteria ... said method is not flagged by a compiler as being synthetic ... adding a tracer to said method only if said method is determined to be complex," as claimed. The Office Action asserts that it is well known that synthetic methods generated by a Java compiler are flagged and thus easily identifiable. Even if this is true, there is no teaching in the prior art **to add a tracer to said method only if said method is determined to be complex**. Thus, there is more missing from the teaching of the prior art than the Office Action concedes. To the extent that the Office Action alleges that it would have been obvious to determine whether a method is complex in order to allow a user to trace the method, the Applicants have already discussed in the response to claim 39 that this allegation amounts to impermissibly using the Applicants' own teaching against them.

Additionally, for reasons discussed in the response to claim 1, Berkley cannot be modified to add a tracer to said method only if said method is determined to be complex for the case in which the criteria for a method being complex is that it is not flagged as being synthetic.

Neither Webster nor Grover remedies the foregoing deficiencies in Berkley.

For the foregoing reasons, claim 47 is patentable.

It is respectfully requested that the rejection of claims 1-3, 5-8, 10, 12-15, 17, 18, 20, 39-42, 44 and 47-54 under 35 U.S.C. §103(a) be withdrawn.

Rejection of Claims 9, 11, 19, 21-24, 26-35, 37, 38, 45 and 46 Under 35 U.S.C. §103(a)

Claims 9, 11, 19, 21-24, 26-35, 37, 38, 45, and 46 have been rejected under 35 U.S.C. §103(a) as being obvious over Berkley in view of Webster in further view of U.S. Patent No. 6,662,359 to Berry (hereinafter "Berry").

Currently amended claim 22 recites:

One or more processor readable storage devices having processor readable code

embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a process comprising:

determining which methods of a set of methods to modify, said step of determining includes determining which methods satisfy a set of one or more criterion for likely being at the top of a call graph; and

modifying for a particular purpose only those methods that satisfy the set of one or more criterion.

Support for the amendment to claim 22 may be found in the instant specification at least in paragraphs 32-35. As described in the instant Specification in paragraph 34, determining which methods are at the top of the call graph can, in some situations, be very difficult. In paragraph 35, an example set of criteria is provided for determining whether the method is likely to be at the top of a call graph.

Applicants respectfully assert that Berkley in view of Webster in further view of Berry fails to disclose, "determining which methods of a set of methods to modify, said step of determining includes determining which methods satisfy a set of one or more criterion for likely being at the top of a call graph," as claimed.

Neither Berkley nor Webster nor Berry, alone or in combination, disclose "determining which methods of a set of methods to modify, said step of determining includes determining which methods satisfy a set of one or more criterion for likely being at the top of a call graph," as claimed. Further, neither Berkley nor Webster, alone or in combination, disclose "modifying for a particular purpose those methods that satisfy the set of one or more criterion," as claimed.

Berkley is concerned with inserting functions into an existing application executable of an object-oriented computer system without recompiling the executable. Berkley modifies runtime configuration settings to add a setting that specifies the function for at least one class of the application executable. Berkley runs the application executable using the modified configuration settings, and if it is determined that a function is active for a class then a re-direction stub is created dynamically to implement the function for the class (col. 2, lines 18-29).

Berkley discusses that in an object oriented system it can be difficult to determine the flow of the program. In this discussion, Berkeley refers to a call graph and the complexities that arise when different methods call each other. Specifically, Berkeley states that if an error occurred in a

"current" method, then the execution of all "previous" methods is important to know the state that an object was in when the current method was called (col. 1, lines 43-62).

What the foregoing clearly indicates is that Berkley is simply not interested in, "determining which methods satisfy a set of one or more criterion for likely being at the top of a call graph," as claimed. Nor would one of ordinary skill in the art have any reason to modify Berkley to determine which methods are at likely to be at the top of the call graph (and modify only those methods). To do so would not provide Berkley with the information such as, ""the state on object was in when the current method was called.

For the foregoing reasons, Berkley does not disclose, and cannot be modified to arrive at, the limitations of claim 22.

The Office Action appears to reason that Webster discloses that a user can specify which methods to trace according to some purpose or interest of the user. Applicants assert that it is not sufficient to allege that it is well known in the art that a method may call another method and therefore it **is of interest to modify the calling method**. That is, just because someone of ordinary skill in the art is aware that one method calls another method does not mean that it is well known in the art that it would be of interest to modify the calling method. In other words, there is a large gap between knowing that a method calls another method and deciding to modify the calling method because it calls another method. Applicants respectfully assert that it is improper to attempt to fill in this gap by taking "Official Notice." Applicants respectfully assert that **the Office Action presents no documentary evidence** to support that it is well known in the computer arts to determine whether to modify a method based on whether or not a method calls another method.

Berry was used as allegedly teaching including a timer for the method. However, Berry does not remedy the foregoing deficiencies in Berkley and Webster.

Claim 33 recites:

> One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a process comprising:
>
> determining whether to trace a method, said step of determining includes determining whether said method calls another method; and

tracing said method for a particular purpose if said method calls another method.

Claim 33 is allowable for at least the reasons discussed in the response to claim 1.

It is respectfully requested that the rejection of claims 9, 11, 19, 21-24, 26-35, 37, 38, 45, and 46 under 35 U.S.C. §103(a) be withdrawn.

New Claims

Claims 55 – 59 have been added. Support for new claim 55 may be found in the instant specification at least in paragraphs 32-37. For example, in paragraph 35, an example is presented of diving methods into a group referred to as complex and a group referred to as simple. Paragraph 35 also discloses that methods may be placed into either a first group or a second group based on access level, whether or not the method is synthetic, and whether the method calls another method.

Support for new claims 56-58 may be found in paragraph 35 of the instant specification. Support for new claim 58 may be found in the instant specification at least in paragraphs 35-36.

New claim 55 recites:

A method, comprising:

dividing a plurality of methods into a first group and a second group, each method is placed into either the first group or the second group based on whether the method calls another method, an access level of the method, and whether the method is synthetic; and

adding monitoring code to each method in the first group but not to any methods in the second group.

The prior art fails to disclose, "dividing a plurality of methods into a first group and a second group, each method is placed into either the first group or the second group based on whether the method calls another method, an access level of the method, and whether the method is synthetic," as claimed.

Further, the prior art fails to disclose, "adding monitoring code to each method in the first group but not to any methods in the second group," as claimed.

New claim 59 depends from claim 22 and recites that, "said determining which methods satisfy a set of one or more criterion for likely being at the top of a call graph is not based on a call graph." Support for new claim 59 may be found in the instant specification in paragraphs 32-37. Note that a technique discloses therein does not require that a call graph is constructed. Rather, a set of one or more criterion is used to determine whether to modify the method. Thus, to the extent that the prior art teaches the use of a call graph for alleged purposes such as determining whether a method calls another method, that prior art is not applicable to claim 59.

Based on the above amendments and these remarks, reconsideration of claims 1-3, 5-15, 17-24, 26-30, 32-35, 37-42 and 44-49, 51-54 and consideration of new claims 55-59 is respectfully requested.

The Examiner's prompt attention to this matter is greatly appreciated. Should further questions remain, the Examiner is invited to contact the undersigned attorney by telephone.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: June 3, 2009

By: /RonaldMPomerenke/
Ronald M. Pomerenke
Reg. No. 43,009

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105-4206
Telephone: (415) 369-9660
Facsimile: (415) 369-9665